OpenWPM

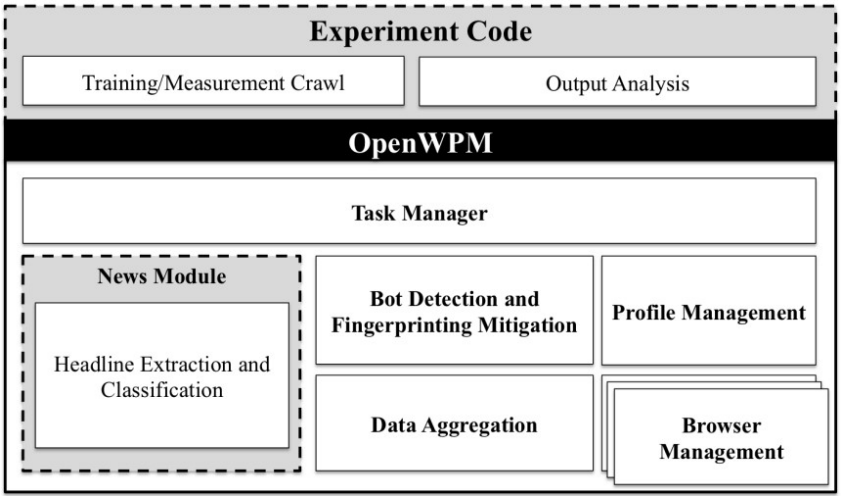
1. OpenWPM (Open Web Privatecy Measurement)

OpenWPM là một phần mềm mã nguồn mở hỗ trợ giám sát, đo lường các trang web phục vụ việc phát hiện, mô tả và định lượng các hành vi ảnh hưởng đến quyền riêng tư (Privacy) của người dùng. Mục tiêu hướng đến gần đây là các dạng mới của Fingerprinting (một cách thức tạo ra một hình ảnh giống với dấu vân tay để theo dõi người dùng, từ đó đưa ra các quảng cáo một cách chính xác hơn). Sau một thời gian dài phát triển, ngày nay WPM còn bao gồm đo lường những vi phạm bảo mật (Security).

2. Hoạt động

OpenWPM hoạt động trên nền Ubuntu 14.04/16.04, sử dụng browser Firefox với ngôn ngữ là python.

OpenWPM cung cấp các cấu trúc modules độc lập để kích hoạt cho mỗi thu thập, các dạng module chính: HTTP Request và Response Headers, redirects, POST request; Javascript Calls; Flash Cookies; Cookie Access; Log Files; Browser Profile; nguồn trang kết suất; Screenshots.



Hình 1: *Code chạy trên OpenWPM giao tiếp với trình quản lý.*

Quá trình một code thử nghiệm chạy trên OpenWPM (*hình 1*), thông qua các block để giao tiếp với trình quản lý: **Task Manager** (phân phối các lệnh từ người dùng đến browser); **Data Aggregation** (nhận dữ liệu, thao tác với dữ liệu nếu cần và ghi dữ liệu thu thập được vào Database); **Bot Dectection** (mô phỏng các hoạt động của người dùng thực như cuộn chuột, di chuyển chuột, delay ngẫu nhiên khi load page để tránh các đe dọa đến kết quả thí nghiệm); **Fingerprinting Mitigation** (ngăn chặn việc theo dõi kiểu Fingerprinting khi chạy nhiều browsers trên cùng một máy bằng cách tạo ra một chuỗi người dùng); **Browser Manger**

(nhận lệnh từ Task Manager, sau đó chuyển đến bộ thực thi lệnh, nhận lệnh tuple và chuyển đôi rnos thành các action của web driver).

3. Mã nguồn

Khai báo địa chỉ các site cần khảo sát tại trường sites = [… , …] và số lượng cửa sổ được bật lên chạy song song và truy cập vào các sites đã được khai báo.

```
NUM_BROWSERS = 3
sites = [‘http://www.example.com’ ,
         ‘http://www.princeton.edu’ ,
         ‘http://www.citp.princeton.edu/’]
```

Khởi tạo giá trị ban đầu của các biến manager_params và browser_params theo các giá trị mặc định của các Browser.

Tiếp sau đó, khởi chạy vòng lặp, cập nhật lại cấu hình của mỗi browser tương ứng mỗi vòng lặp. Thông tin cấu hình cần cập nhật lại bao gồm: Cho phép ghi lại các trường HTTP Request và HTTP Respond (http_instrument = true) và Cho phép các trình duyệt bật flash plugin (disable_flash = false).

```
manager_params, browser_params =
TaskManager.load_default_params(NUM_BROWSERS)
for i in range(NUM_BROWSERS):
   browser_params[i]['http_instrument'] = True
   browser_params[i]['disable_flash'] = False
browser_params[0]['headless'] = True
manager_params['data_directory'] = '~/Desktop/'
manager_params['log_directory'] = '~/Desktop/'
```

Chạy các sites đã được khai báo với số trình duyệt NUM_BROWSERS cùng một lúc. Khởi chạy vòng lặp cho các sites, với mỗi một site trong sites , thực thi bắt đầu truy cập trang với thời gian timeout = 60 giây, thời gian sleep = 0; ghi lại các trạng thái cookies; đóng tab hiện tại và đồng bộ giữa các trình duyệt.

```
manager        =        TaskManager.TaskManager(manager_params,
browser_params)
for site in sites:
   command_sequence = CommandSequence.CommandSequence(site)
   command_sequence.get(sleep=0, timeout=60)
   command_sequence.dump_profile_cookies(120)
   manager.execute_command_sequence(command_sequence,
index='**')
```

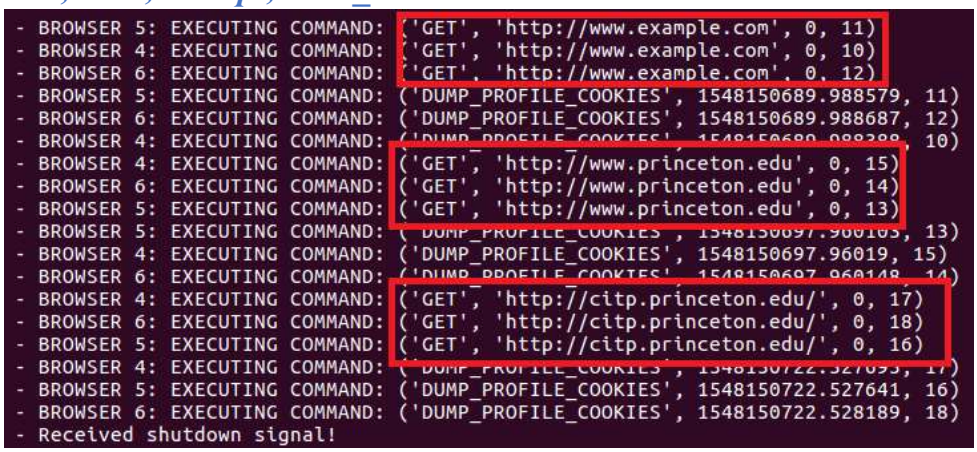
4. Demo

4.1 Các trường dữ liệu thu thập được khi chạy demo:

```
manager.execute_command_sequence(command_sequence, index='**')
```

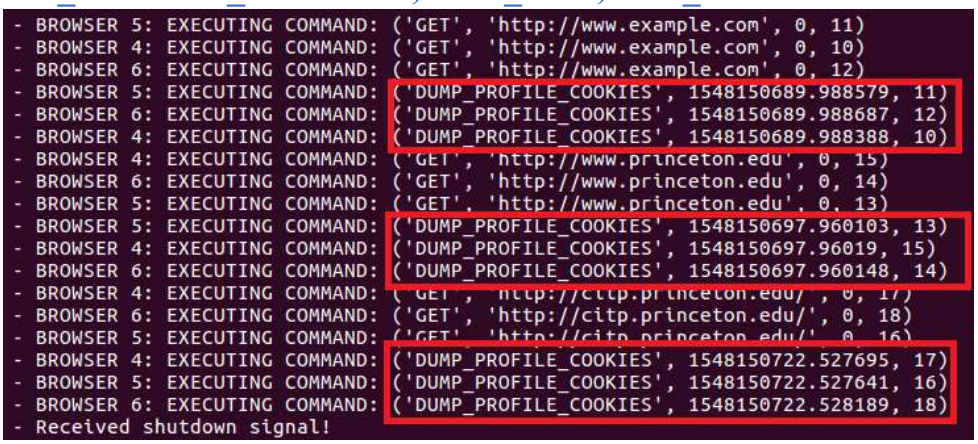
Sẽ có ba browser, được mở song song và truy cập một cách đồng bộ vào http://www.example.com, http://www.princeton.edu, http://citp.princeton.edu/ và các trường dữ liệu thu thập được là

1. ***'GET', 'url', 'sleep', 'visit_id'***



Với lệnh GET của CommandSequence, trình duyệt sẽ thực hiện một lượt truy cập vào trường url với thời gian nghỉ là sleep, visit_id và nội dung các bảng cookie, flash_cookies, http_request, http_response, localStorage và site_visits sẽ được sửa đổi trong Database.

2. ***'DUMP_PROFILE_COOKIES', 'start_time', 'visit_id'***



Với lệnh DUMP_PROFILE_COOKIES của CommandSequence sẽ đọc và lưu mọi thay đổi về trạng thái do url của CommandSequence được lấy từ lệnh GET, sau đó đóng tab của trình duyệt.

4.2 Dữ liệu thu lại được trong database

Database thu được sau khi đo lường trong file **crawl-data.sqlite** bao gồm các table với các trường sau:

‘crawl’: lưu lại các trường *crawl_id* | *task_id* | *browser_params*

‘crawl_history’: lưu lại lịch sử truy cập, tương tự file log trên terminal

‘task’: lưu lại các trường *crawl_id* | *task_id* | *browser_params*

‘http_redirects’: lưu lại các trường *old_channel_id* | *channel_id* của http request

‘http_requests’: lưu lại các trường *id* | *crawl_id* | *visit_id* | *url* | *top_level_url* |*method* | *referrer* | *headers* | *is_XHR* | *is_frame_load* | *is_full_page* | *is_third_party_channel* | *is_third _party_window* | *triggering_origin* | *loading_href* | *req_call_stack* | *content_policy_type* | *post_body* | *channel_id* | *time_stamp*

‘http_responses’: lưu lại các trường khớp với **http_requests**, ở đây trường channel_id được sử dụng để liên kết các request trong bảng http_request với response tương ứng trong bảng **http_responses**

‘javascript’: lưu lại tất cả các hành động calls và quyền truy cập cho các API có khả năng là các Fingerprinting

‘javascript_cookies’: ghi lại các cookie được thiết lập bởi Javascript thông qua HTTP Responses

‘profile_cookies’: thường chứa các cookie được thêm bởi Javascript và HTTP Responses

‘site_visits’: lưu lại các địa chỉ đã truy cập với các trường *id* | *url*